TURBO-VGA640480 使用说明

TURBO-VGA640480 是一块用于普通单片机（包括 51、AVR、PIC、ARM、MSP430 等等）控制 VGA 显示器（包括液晶显示器和 CRT 显示器，只要具备标准 VGA 接口）显示各种图形的控制板。

◆其分辨率为 640X480,色深为 16 位色，能显示 65536 种颜色
◆本 VGA 卡具备 4 屏缓存区，能根据指令很快地换屏显示。
◆采用 8 位并行总线进行图像图形数据传送，速度快，
◆具有硬件的清屏和区域填充指令，能指定清屏后的背景颜色，能对任意位置、任意大小的区域进行颜色数据复制，让很慢的单片机也可以很快的进行显示操作，，
◆与 CPU 的接口兼容 5V 和 3.3V。
◆如果想在显示器屏幕某个位置显示某种颜色，只要向控制板写入 X 坐标（０～６３９，分两次 8 位写入），Y 坐标(0~439, 分两次 8 位写入) ,然后写入 16 位的颜色（分两次 8 位写入）,就可以马上在屏幕上指定的一点看到所需的颜色，
◆通过各种函数就可以实现画线，绘制图片等功能。

控制板与 MCU 的接口如下（从控制板的顶层向下看）：

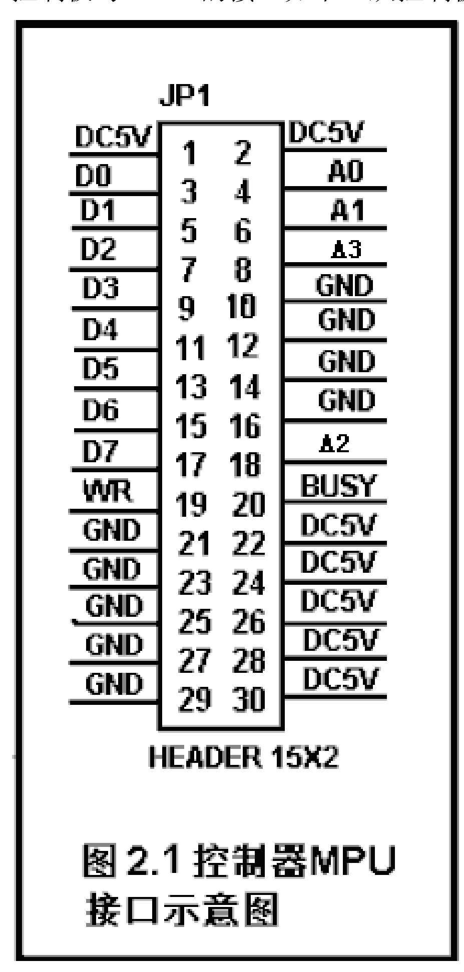


图 2.1 控制器MPU 接口示意图

接口信号定义如下:

符号	管脚	类型	说明
VCC	1,2,22,24,26,28,30	电源5V	直流 5V,确保电源纹波较少
GND	10,12,14,16,21,23,25,27,29	地	地线
D0~D7	3,5,7,9,11,13,15,17	I	数据总线,图像图形数据通过该端口写入
BUSY	20	O	忙线:高电平有效，表示控制器正在进行内部的数据写入，这时外部 MCU 不能向控制器写入数据
WR	19	I	写数据线，上下跳变都有效，当 WR 从高到低跳变时，D0~D7 被写入控制器低位，当 WR 从低到高跳变时，D0~D7 被写入控制器的高位
A0~A3	4,6,8,18	I	地址线，用于区分写入 X 坐标，Y 坐标，和颜色，当 A1A0=00 时，数据总线上的数据写进控制器的颜色寄存器，当 A1A0=01 时，数据总线上的数据写进 X 坐标，当 A1A0=10 时，数据总线上的数据写进 Y 坐标

信号真值表

WR	A3	A2	A1	A0	操作
↓	0	0	0	0	数据总线→颜色寄存器(低 8 位)
↑	0	0	0	0	数据总线→颜色寄存器(高 8 位)
↓	0	0	0	1	数据总线→X 坐标寄存器(低 8 位) (1~640)
↑	0	0	0	1	数据总线→X 坐标寄存器(高 2 位) (1~640)
↓	0	0	1	0	数据总线→Y 坐标寄存器(低 8 位)(1~480)
↑	0	0	1	0	数据总线→Y 坐标寄存器(高 1 位)(1~480)
↓ ↑	0	0	1	1	写显示配置寄存器(见显示配置寄存器说明)
↓	0	1	0	0	写复制寄存器 源地址 X 坐标(低 8 位)
↑	0	1	0	0	写复制寄存器 源地址 X 坐标(高 2 位)
↓	0	1	0	1	写复制寄存器 源地址 Y 坐标(低 8 位)
↑	0	1	0	1	写复制寄存器 源地址 Y 坐标(高 1 位)
↓	0	1	1	0	写复制寄存器 目标地址 X 坐标(低 8 位)
↑	0	1	1	0	写复制寄存器 目标地址 X 坐标(高 2 位)
↓	0	1	1	1	写复制寄存器 目标地址 Y 坐标(低 8 位)
↑	0	1	1	1	写复制寄存器 目标地址 Y 坐标(高 1 位)
↓	1	0	0	0	写复制寄存器 复制区域宽度(低 8 位)
↑	1	0	0	0	写复制寄存器 复制区域宽度(高 2 位)
↓	1	0	0	1	写复制寄存器 复制区域高度(低 8 位)
↑	1	0	0	1	写复制寄存器 复制区域高度(高 1 位)
↓ ↑	1	0	1	0	写复制配置寄存器(见复制配置寄存器说明)
↓ ↑	1	0	1	1	写 执行复制 或 执行区域填充 命令寄存器
↓	1	1	0	0	写填充颜色寄存器（低 8 位）
↑	1	1	0	0	写填充颜色寄存器（高 8 位）

显示配置寄存器如下表:

D7	D6	D5	D4	D3	D2	D1	D0
保留位	保留位	保留位	保留位	当前写入区选择位 1	当前写入区选择位 0	当前显示区选择位 1	当前显示区选择位 0
				默认为 0	默认为 0	默认为 0	默认为 0

注意：本卡共有 4 个区可供选择进行显示，写入的操作区可以跟显示区相同，也可以不同，当相同时，写入的数据会马上显示出来，当不相同时写入的数据不会马上显示出来，而是当你写好数据后，把当前显示区域设置到你刚刚写进的区域，就可以看到刚才写进的图形。

复制配置寄存器如下表:

D7	D6	D5	D4	D3	D2	D1	D0
保留位	保留位	保留位	保留位	复制目标区选择位 1	复制目标区选择位 0	复制源区选择位 1	复制源区选择位 0
				默认为 0	默认为 0	默认为 0	默认为 0

⑴、“复制源区”代表从哪个区复制
⑵、“复制目标区”代表数据要复制到那个区
⑶、共有 4 个区，(位 1、位 0)=00 时为第一个区，(位 1、位 0)=01 时为第二个区，(位 1、位 0)=10 时为第三个区，(位 1、位 0)=11 时为第四个区

复制源地址 X 坐标寄存器(低 8 位)

D7	D6	D5	D4	D3	D2	D1	D0
地址位 7	地址位 6	地址位 5	地址位 4	地址位 3	地址位 2	地址位 1	地址位 0
默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0

复制源地址 X 坐标寄存器(高 2 位)

D7	D6	D5	D4	D3	D2	D1	D0
保留位	保留位	保留位	保留位	保留位	保留位	地址位 9	地址位 8
默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0

复制源地址 Y 坐标寄存器(低 8 位)

D7	D6	D5	D4	D3	D2	D1	D0
地址位 7	地址位 6	地址位 5	地址位 4	地址位 3	地址位 2	地址位 1	地址位 0
默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0

复制源地址 Y 坐标寄存器(高 1 位)

D7	D6	D5	D4	D3	D2	D1	D0
保留位	保留位	保留位	保留位	保留位	保留位	保留位	地址位 8
默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0

注:复制过程的数据来自于复制源地址 X 坐标寄存器、复制源地址 Y 坐标寄存器与复制配置寄存器中的”复制源区选择位”;譬如你要从 1 区的 X(10)，Y(20)的位置复制数据，就要设定复制源地址 X 坐标寄存器=10，复制源地址 Y 坐标寄存器=20，”复制源区选择位”=(0,1)。

复制目标地址 X 坐标寄存器(低 8 位)

D7	D6	D5	D4	D3	D2	D1	D0
地址位 7	地址位 6	地址位 5	地址位 4	地址位 3	地址位 2	地址位 1	地址位 0
默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0

复制目标地址 X 坐标寄存器(高 2 位)

D7	D6	D5	D4	D3	D2	D1	D0
保留位	保留位	保留位	保留位	保留位	保留位	地址位 9	地址位 8
默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0

复制目标地址 Y 坐标寄存器(低 8 位)

D7	D6	D5	D4	D3	D2	D1	D0
地址位 7	地址位 6	地址位 5	地址位 4	地址位 3	地址位 2	地址位 1	地址位 0
默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0

复制目标地址 Y 坐标寄存器(高 1 位)

D7	D6	D5	D4	D3	D2	D1	D0
保留位	保留位	保留位	保留位	保留位	保留位	保留位	地址位 8
默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0	默认为 0

注:复制过程的数据将会写进复制目标地址 X 坐标寄存器、复制目标地址 Y 坐标寄存器与复制配置寄存器中的”复制目标区选择位”;譬如你要将数据写进 2 区的 X(100)，Y(200)的位置复制数据，就要设定

复制目标地址 X 坐标寄存器=100，复制目标地址 Y 坐标寄存器=200，”复制目标区选择位”=(1,0)。

执行复制 或 执行区域填充 命令寄存器

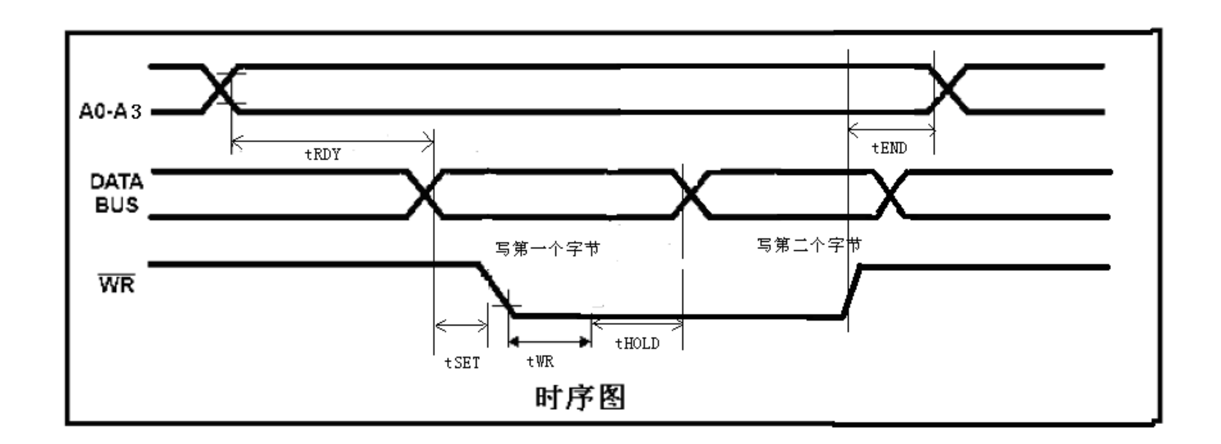
D7	D6	D5	D4	D3	D2	D1	D0
保留位	保留位	保留位	保留位	保留位	保留位	写 1 时开始执行区域填充	写 1 时开始执行数据复制
						默认为 0	默认为 0

(1)、区域填充的颜色为“填充颜色寄存器”

(2)、区域填充命令设定的填充区域为：坐标（目标地址 X 寄存器，目标地址 Y 寄存器），宽度（复制区域宽度寄存器）,高度(复制区域高度寄存器)

时间与时序图

	最小	典型	最大
tRDY	10ns		
tSET	10ns		
tWR	30ns		
tHOLD	20ns		
tEND	10ns		



时序图

编程方法

A:写入 X 坐标（A3=0,A2=0,A1=0,A0=1）

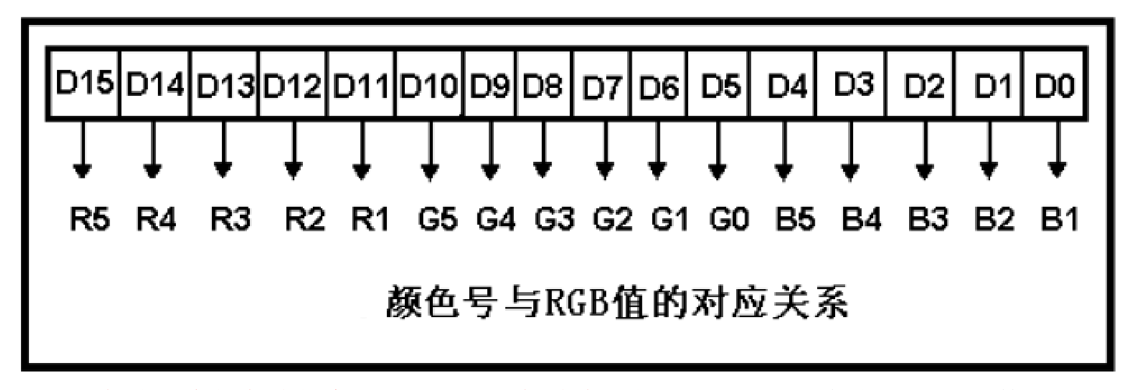
X 的范围时 1 到 640，需要 10 位，第一个字节写入低八位(D7,D0)→(X7,X0)，第二个字节写入高两位(D1,D0)→(X9,X8)。

B:写入 Y 坐标（A3=0,A2=0,A1=1,A0=0）

Y 的范围时 1 到 480，需要 9 位，第一个字节写入低八位，第二个字节写入高一位(D0)→(Y8)。

C:写入颜色值（A3=0,A2=0,A1=0,A0=0）

颜色值是 16 位的，共有 65536 种颜色，第一次写入 16 位中的低八位，第二次写入高八位，颜色值分配采用 R5G6B5 的方法，如下表：



颜色号与RGB值的对应关系

另外，每次写完颜色值的高八位后，X 坐标值自动加一，以便于更快的写入图形数据。

如果 MCU 的操作速度非常快（一般 8 位单片机达不到），在操作控制前应查询 BUSY 信号是否为高，如果 BUSY=1，则不能操作。

D、设置显示区与写入区

本卡显示缓冲分成 4 个区，显示区可以选择其中一个区作为显示区，也可以选择其中一个区作为写入区，显示区和写入区可以相同，也可以不同，当显示区和写入区相同时，如大家都是 1 区时(这时“显示配置寄存器”写入的值为 00000101(0x05)),写入的图形数据会在屏幕上显示出来，相反，如果显示区和写入区不同时，如显示区为 0，写入区为 1（这时“显示配置寄存器”写入的值为 00000100（0x04））,这时写进的数据不会马上显示出来，只是写进了缓冲，你可以在在写完缓冲数据后，再把显示区切换到缓冲区，这样图形的显示不会因为你的单片机速度慢而造成用户会看到图形的刷新过程。

E、图形数据的复制（图形区域的复制）

该功能是用于在显存的不同区域进行图形数据的复制，由本显示卡硬件完成的，目的是让速度慢的单片机也可以产生很流畅的效果，如当你的界面上有一个按键的图形，平时该按键图形显示为松开的状态，当你按下的时候，这个按键的图形应该要显示成按下的状态，如大家熟知的 开始（松开状态），开始(按下状态)

这样就要比较快的更换该区域的图形数据，如果用没有复制功能的显示卡，这个过程要用单片机来完成，但是当单片机速度比较慢的时候，图形的刷新过程会比较慢，这样看起来效果会比较差。如果使用复制功能，你可以先把按键按下和松开的图形数据写进缓冲区，然后用一个命令就可以复制不同的图形到该区域，相当于很快的更换的该区域的图形。

F、区域填充功能(当填充的区域为全屏时，就是清屏功能)

该功能用于快速完成某个区域的颜色填充,只需指定要填充区域的左上角 XY 坐标，指定填充的宽度，高度，指定填充的颜色，然后启动填充功能，颜色填充就由硬件完成了，这个过程 BUSY 线会被拉高，当 BUSY 线拉低后，颜色填充表示已经完成了，你可以对板卡进行别的操作了。如果填充区域的 XY 坐标设成 1，1，然后宽度设成 640，高度设成 480，区域填充就相当于清屏功能了。

VGA 屏幕的坐标系：

（1，1）　　　　　　　　　　　　　　　　（640，1）

（1，480）　　　　　　　　　　　　　　（640，480）

以下子程序是在 16MHz　AVR 单片机中运行的,程序中去掉检测 BUSY 信号。

写 X 坐标子程序

```
void write_lcd_x_location(unsigned int wr_dat)
{;
   HI_LCD_A0;
   LO_LCD_A1;
   LO_LCD_A2;
   LO_LCD_A3;

   LCD_PORT=(wr_dat&0x00ff);              //写低八位
   asm("nop");
   LO_LCD_WR;

   LCD_PORT=((wr_dat>>8)&0x0003);      //写高两位
   asm("nop");
   HI_LCD_WR;
}
```

写 Y 坐标子程序

```
void write_lcd_y_location(unsigned int wr_dat)
{
     LO_LCD_A0;
   HI_LCD_A1;
   LO_LCD_A2;
   LO_LCD_A3;
```

```
   LCD_PORT=(wr_dat&0x00ff);       //写低八位
   asm("nop");
   LO_LCD_WR;

   LCD_PORT=((wr_dat>>8)&0x0001);       //写高 1 位
   asm("nop");
   HI_LCD_WR;


}
```

写配置寄存器

```
void write_config_register(unsigned int wr_dat)
{
    HI _LCD_A0;
   HI_LCD_A1;
   LO_LCD_A2;
   LO_LCD_A3;

   LCD_PORT=(wr_dat&0x00ff);       //写八位配置值
   asm("nop");
   LO_LCD_WR;
   asm("nop");
   HI_LCD_WR;
}
```

写颜色值（只有写完颜色值显示屏才会显示）

```
void write_lcd_color(unsigned int wr_dat)
{
    LO_LCD_A0;
   LO_LCD_A1;
   LO_LCD_A2;
   LO_LCD_A3;

   LCD_PORT=(wr_dat&0x00ff);                //写低八位
   asm("nop");
   LO_LCD_WR;


   LCD_PORT=((wr_dat>>8)&0x00ff);              //写高八位
   asm("nop");
   HI_LCD_WR;
}
```

在显示屏上 0 页用指定颜色写入一个点，同时设置 0 页为显示页，子程序

```
void set_point_color(unsigned int x,unsigned int y,unsigned int color)
{
   write_config_register(0x00);
   write_lcd_x_location(x);
   write_lcd_y_location(y);
   write_lcd_color(color);       //写完后 X 自动加 1
}
```

设置显示配置寄存器

```
/*参数说明
      display_area:当前正在 VGA 显示器显示的区域
      write_area:当前通过总线写进的区域
*/
void set_display_config_register(unsigned char display_area,unsigned char write_area)
{
      unsigned char tmp_wr;

      tmp_wr=0;

      //////////////////////处理显示区
      if(display_area==1)
      {
            tmp_wr|=0x01;
      }
      else if(display_area==2)
      {
            tmp_wr|=0x02;
      }
      else if(display_area==3)
      {
            tmp_wr|=0x03;
      }

      ///////////////////////////////处理写入区
      if(write_area ==1)
      {
            tmp_wr|=0x04;
      }
      else if(write_area ==2)
      {
            tmp_wr|=0x08;
```

```
        }
        else if(write_area==3)
        {
                tmp_wr|=0x0C;
        }


        HI_LCD_A0;
    HI_LCD_A1;
    LO_LCD_A2;
    LO_LCD_A3;

    LCD_PORT= tmp_wr;
    asm("nop");
    LO_LCD_WR;
    asm("nop");
    HI_LCD_WR;

}
```

设置"复制寄存器源地址"

```
/*参数说明
        copy_source_addr_x: 即将要被复制的区域的 X 坐标
        copy_source_addr_y: 即将要被复制的区域的 Y 坐标
*/
void set_copy_source_address(unsigned int copy_source_addr_x,unsigned int copy_source_addr_y)
{

        ////////////////////////////////////    写 X
      LO_LCD_A0;
    LO_LCD_A1;
    HI_LCD_A2;
    LO_LCD_A3;

    LCD_PORT=( copy_source_addr_x &0x00ff);                    //写低八位
    asm("nop");
    LO_LCD_WR;


    LCD_PORT=(( copy_source_addr_x >>8)&0x0003);               //写高 2 位
    asm("nop");
    HI_LCD_WR;
```

```
///////////////////////////////////////写 Y

   HI_LCD_A0;
  LO_LCD_A1;
  HI_LCD_A2;
  LO_LCD_A3;

  LCD_PORT=( copy_source_addr_y &0x00ff);          //写低八位
  asm("nop");
  LO_LCD_WR;


  LCD_PORT=(( copy_source_addr_y >>8)&0x0001);          //写高 1 位
  asm("nop");
  HI_LCD_WR;

}
```

设置”复制寄存器目标地址”

```
/*参数说明
    copy_target_addr_x:  即将复制(或填充)到的区域的 X 坐标
    copy_target_addr_y:  即将复制(或填充)到的区域的 Y 坐标
*/
void set_copy_target_address(unsigned int copy_target_addr_x,unsigned int copy_target_addr_y)
{

    /////////////////////////////////   写 X
   LO_LCD_A0;
  HI_LCD_A1;
  HI_LCD_A2;
  LO_LCD_A3;

  LCD_PORT=( copy_ target _addr_x &0x00ff);          //写低八位
  asm("nop");
  LO_LCD_WR;


  LCD_PORT=(( copy_ target _addr_x >>8)&0x0003);          //写高 2 位
  asm("nop");
  HI_LCD_WR;

///////////////////////////////////////写 Y
```

```
  HI_LCD_A0;
 HI_LCD_A1;
 HI_LCD_A2;
 LO_LCD_A3;

 LCD_PORT=( copy_source_addr_y &0x00ff);            //写低八位
 asm("nop");
 LO_LCD_WR;


 LCD_PORT=(( copy_source_addr_y >>8)&0x0001);          //写高 1 位
 asm("nop");
 HI_LCD_WR;

}
```

设置复制区域宽度寄存器和高度寄存器

```
/*参数说明
    area_width:被复制(或填充)区域的宽度
    area_height:被复制(或填充)的区域的高度
*/
void set_copy _width_height(unsigned int area_width,unsigned int area_height)
{

    ////////////////////////////////////   写宽度
  LO_LCD_A0;
 LO_LCD_A1;
 LO_LCD_A2;
 HI_LCD_A3;

 LCD_PORT=( area_width &0x00ff);            //写低八位
 asm("nop");
 LO_LCD_WR;


 LCD_PORT=(( area_width >>8)&0x0003);          //写高 2 位
 asm("nop");
 HI_LCD_WR;

/////////////////////////////////////////////写高度

  HI_LCD_A0;
 LO_LCD_A1;
 LO_LCD_A2;
```

```
  HI_LCD_A3;

  LCD_PORT=( area_height &0x00ff);          //写低八位
  asm("nop");
  LO_LCD_WR;


  LCD_PORT=(( area_height >>8)&0x0001);          //写高 1 位
  asm("nop");
  HI_LCD_WR;

}
```

设置复制配置寄存器

```
/*参数说明
	source_area:被复制(或填充)区域所属区(0~3)
	target_area:复制(或填充)到的区域所属区(0~3)
*/
void set_copy_config_register(unsigned char source_area,unsigned char target_area)
{
	unsigned char tmp_wr;

	tmp_wr=0;

	//////////////////////处理被复制区域
	if(source_area ==1)
	{
		tmp_wr|=0x01;
	}
	else if(source_area ==2)
	{
		tmp_wr|=0x02;
	}
	else if(source_area ==3)
	{
		tmp_wr|=0x03;
	}

	//////////////////////////////处理复制到的区域
	if(target_area ==1)
	{
		tmp_wr|=0x04;
	}
	else if(target_area ==2)
```

```
    {
        tmp_wr|=0x08;
    }
    else if(target_area ==3)
    {
        tmp_wr|=0x0C;
    }


    LO_LCD_A0;
  HI_LCD_A1;
  LO_LCD_A2;
  HI_LCD_A3;

  LCD_PORT= tmp_wr;
  asm("nop");
  LO_LCD_WR;
  asm("nop");
  HI_LCD_WR;

}
```

设置填充颜色寄存器

```
/*参数说明
    fill_color:填充颜色值
*/
void set_fill_color(unsigned int fill_color)
{

    ////////////////////////////////
   LO_LCD_A0;
  LO_LCD_A1;
  HI_LCD_A2;
  HI_LCD_A3;

  LCD_PORT=( fill_color &0x00ff);                //写低八位
  asm("nop");
  LO_LCD_WR;


  LCD_PORT=(( fill_color >>8)&0x0003);            //写高 2 位
  asm("nop");
  HI_LCD_WR;
```

```
}
```

设置执行寄存器

```
/*参数说明
        do_what:要执行的命令值(1:执行复制,2:执行填充)
*/
void set_action_register(unsigned char do_what)
{

        /////////////////////////////////////
      HI_LCD_A0;
    HI_LCD_A1;
    LO_LCD_A2;
    HI_LCD_A3;

    LCD_PORT=do_what;
    asm("nop");
    LO_LCD_WR;
    asm("nop");
    HI_LCD_WR;

}
```

用指定颜色请屏子程序(软件方法)

```
void clear_screen_by_software(unsigned int color)
{
    unsigned int i,j;
    for(i=1;i<=480;i++)
    {
        for(j=1;j<=640;j++)
        {
            write_lcd_color(color);
        }
        write_lcd_y_location(i);
        write_lcd_x_location(0);
    }
}
```

用指定颜色请屏子程序(硬件方法)

```
void clear_screen_by_hardware(unsigned int color)
{
```

```
        set_copy_target_address(0,0);   //设置目标地址坐标
        set_copy _width_height(640,480);   //设置填充宽度和高度
        set_fill_color(color);   //设置填充颜色
        set_copy_config_register(0,0);      //设置要填充的区为 0 （共有 4 个区）
        set_action_register(2);             //开始填充
  while(busy==1){} //此处请等待命令执行完，如果是填充整屏（清屏）时间大概 1/60 秒

}
```

从 1 区整屏复制到 0 区(硬件方法)

```
void copy_screen_by_hardware(void)
{
        set_copy_source_address(0,0)     //设置源地址坐标
        set_copy_target_address(0,0);   //设置目标地址坐标
        set_copy _width_height(640,480);   //设置填充宽度和高度
        set_copy_config_register(1,0);      //设置源复制区为 1 区,目标区为 0 区
        set_action_register(1);             //开始复制
  while(busy==1){} //此处请等待命令执行完，如果是填充整屏（清屏）时间大概 1/60 秒

}
```

```
//画圆函数
void circle(int centerx, int centery, int radius, int color, int type)
{
  int x = 0;
  int y = radius;
  int delta = 2*(1-radius);
  int direction;
  while (y >= 0) {
    if (!type) {
      setPixel(centerx+x, centery+y, color);
      setPixel(centerx-x, centery+y, color);
      setPixel(centerx-x, centery-y, color);
      setPixel(centerx+x, centery-y, color);
    }
    else {
      line_b(centerx+x, centery+y, centerx+x, centery-y, color, 1);
      line_b(centerx-x, centery+y, centerx-x, centery-y, color, 1);
    }
    if (delta < 0) {
      if ((2*(delta+y)-1) < 0) {
        direction = 1;
```

```
    }
    else {
      direction = 2;
    }
  }
  else if(delta > 0) {
    if ((2*(delta-x)-1) <= 0) {
      direction = 2;
    }
    else {
      direction = 3;
    }
  }
  else {
    direction=2;
  }

  switch(direction) {
  case 1:
    x++;
    delta += (2*x+1);
    break;
  case 2:
    x++;
    y--;
    delta += 2*(x-y+1);
    break;
  case 3:
    y--;
    delta += (-2*y+1);
    break;
  }

 }
}
```

```
//画线函数
void    cb_line(int    x1,int    y1,int    x2,int    y2)      /*draw    a    line*/
  {
          int    dx,dy,n,k,i,f;
          int    x,y;
          dx=abs(x2-x1);
```

```
dy=abs(y2-y1);
n=dx+dy;
if(x2>=x1)
{
        k=y2>=y1?1:4;
        x=x1;
        y=y1;                                 /*    k    stands    for    the
slope    of    line*/
}
else
{
        k=y2>=y1?2:3;
        x=x2;
        y=y2;
}

for(i=0,f=0;i<n;i++)
      if(f>=0)
      switch(k)
      {
              case    1:putpixel(x++,y,3);f-=dy;break;
              case    2:putpixel(x,y++,3);f-=dx;break;
              case    3:putpixel(x--,y,3);f-=dy;break;
              case    4:putpixel(x,y--,3);f-=dx;break;
        }
      else
      switch(k)
      {
              case    1:putpixel(x,y++,3);f+=dx;break;
              case    2:putpixel(x--,y,3);f+=dy;break;
              case    3:putpixel(x,y--,3);f+=dx;break;
              case    4:putpixel(x++,y,3);f+=dy;break;
      }
}
```

汇编

```
LCD_A0      EQU     P1.0
LCD_A1      EQU     P1.1
LCD_A2      EQU     P1.2
```

```
LCD_A3      EQU     P1.3
LCD_WR      EQU     P1.4

LCD_PORT        EQU    P2

;R0  为数据低位
;R1  为数据高位
```

写 X 坐标子程序

```
write_lcd_x_location:
   SETB       LCD_A0
   CLR        LCD_A1
   CLR        LCD_A2
   CLR        LCD_A3

   MOV      A ,R0
   MOV       LCD_PORT, A
   NOP
   CLR      LCD_WR

   MOV     A ,R1
   MOV     LCD_PORT ,A
   NOP
   SETB   LCD_WR

   RET
```

写 Y 坐标子程序

```
write_lcd_y_location:
   CLR        LCD_A0
   SETB       LCD_A1
   CLR        LCD_A2
   CLR        LCD_A3

   MOV      A ,R0
   MOV       LCD_PORT, A
   NOP
   CLR      LCD_WR

   MOV     A ,R1
   MOV     LCD_PORT ,A
   NOP
```

```
  SETB   LCD_WR

  RET
```

写配置寄存器

```
write_config_register:
  SETB        LCD_A0
  SETB        LCD_A1
  CLR         LCD_A2
  CLR         LCD_A3

  MOV       A ,R0
  MOV       LCD_PORT, A        ;写八位配置值
  NOP
  CLR       LCD_WR
  NOP
  SETB   LCD_WR

  RET
```

写颜色值（只有写完颜色值显示屏才会显示）

```
write_lcd_color:

  CLR         LCD_A0
  CLR         LCD_A1
  CLR         LCD_A2
  CLR         LCD_A3

  MOV     A, R0
  MOV       LCD_PORT, A
  NOP
  CLR       LCD_WR

  MOV     A ,R1
  MOV     LCD_PORT, A
  NOP
  SETB   LCD_WR

  RET
```

在显示屏上 0 页用白色写入坐标 1，1 一个点，同时设置 0 页为显示页，子程序

```
void set_point_color(unsigned int x,unsigned int y,unsigned int color)
{

 MOV     R0,#00H
 MOV     R1,#00H
 LCALL   write_config_register

 MOV     R0,#01H
 MOV     R1,#00H
 LCALL   write_lcd_x_location

 MOV     R0,#01H
 MOV     R1,#00H
 LCALL   write_lcd_y_location

 MOV     R0,#0FFH
 MOV     R1,#0FFH
 LCALL   write_lcd_color    ; 写完后 X 自动加 1

}
```

一般图形取模工具用 Image2Lcd

而一般的字符取模工具用 zimo221